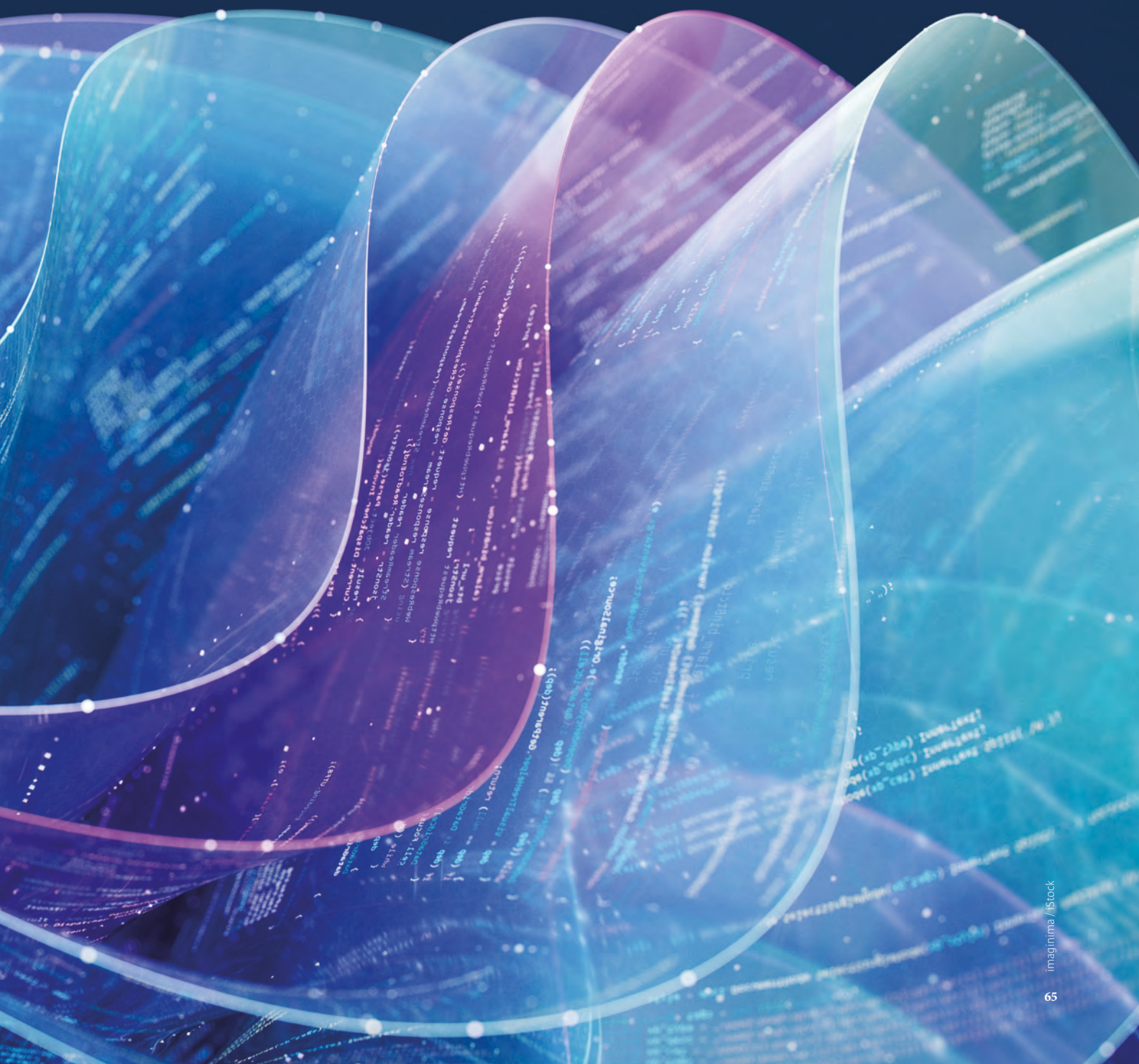
Nörosembolik Yapay Zekâ

Dr. Mahir E. Ocak [*TÜBİTAK Bilim ve Teknik Dergisi*

Yapay zekâ uygulamalarının geliştirilmesinde kullanılan iki ana programlama stili var: sembolik programlama ve yapay sinir ağları. Uzun yıllar boyunca sembolik yapay zekâ ön plandaydı. Son 10-20 yılda ise yapay sinir ağlarıyla modellenen uygulamalar önemli başarılar elde etti. Birkaç yıldır üzerine yoğun araştırmalar yapılan bir konu ise bu iki programlama stilinin bir arada kullanılması. Pek çok araştırmacı yapay genel zekâya ulaşmanın ancak sembolik algoritmalar ile yapay sinir ağlarını bir araya getiren nörosembolik yaklaşımla mümkün olabileceğini düşünüyor.



imaginima / iStock

Just_Super / iStock

Sembolik Yapay Zekâ

Sembolik yapay zekâ uygulamalarında insan zekâsı gerektiren karmaşık problemleri çözmek için sembollerden ve sembolik mantıktan yararlanılır.

Sembolik yapay zekâ uygulamalarının temel bileşenlerinden biri, bilginin sembolik bir biçimde (insanların okuyup anlayabileceği bir biçimde) kodlanmasıdır. Bu amaçla kullanılan çeşitli yöntemler vardır. Örneğin mantık programlaması olarak adlandırılan yöntemde bilgi mantıksal ifadeler ve kurallarla kodlanır. Mantık programlaması için kullanılan Prolog, ASP ve Datalog gibi dillerin tamamında kuralları ifade etmek için şu sentaks kullanılır:

$A: -B_1,\ldots,B_n.$

A, kuralın "başı", $B_1,\ldots,B_n$ ise kuralın "gövdesi" olarak adlandırılır. Bu sentaks, kuralın gövdesinde yer alan B_i "koşulları" doğruysa A'nın doğru olduğunu ifade eder. Eğer bir kuralın doğruluğu herhangi bir koşula bağlı değilse kural basitçe

$A.$

biçiminde yazılır. Koşulsuz olarak doğru olan A ifadesi bir "gerçeği" kodlar. Kurallarda yer alan $A, B_1,\ldots,B_n$ terimleri $p(t_1,\ldots,t_n)$ biçiminde "atomik formüller" ile ifade edilir. Bu formüllerde p bir ilişkiyi, t_i terimleri ise aralarında ilişki bulunan nesneleri ifade eder. Terimlerde yer alan ifadeler sabit ya da X, Y, Z gibi büyük harflerle ifade edilen değişkenler olabilir. Yapay zekâ uygulamasına bir soru sormak için kullanılan sentaks ise şudur:

$?- B_1,\ldots,B_n.$

Konunun daha iyi anlaşılması için bir örnek ele alalım:

```
baba_cocuk(ahmet, mehmet).
baba_cocuk(mehmet, ali).
anne_cocuk(ayşe, ali).
dede_torun(X, Y) :—
    baba_cocuk(X, Z),
    baba_cocuk(Z, Y).
dede_torun(X, Y) :—
    baba_cocuk(X, Z),
    anne_cocuk(Z, Y).
```

Yukarıdaki kod dizisinin ilk üç satırında yer alan "*A*." biçimindeki koşulsuz ifadeler gerçekleri ifade eder. İlk satırdaki ifade Ahmet ile Mehmet arasında baba-çocuk ilişkisi olduğunu yani Ahmet'in Mehmet'in babası olduğunu söyler. Benzer biçimde ikinci satırdaki ifade Mehmet ile Ali arasında baba-çocuk ilişkisi olduğunu, üçüncü satırdaki ifade ise Ayşe ile Ali arasında anne-çocuk ilişkisi olduğunu söyler. Dördüncü ve yedinci satırda başlayıp üç satıra yayılan kurallar, iki nesne arasında dede-torun ilişkisi olması için gerekli koşulları ifade eder. Bu kurallarda yer alan büyük harflerle yazılmış *X, Y, Z* herhangi bir değer alabilecek değişkenlerdir. İlk kurala göre *X*'in çocuğu ve *Y*'nin babası olan bir *Z* kişisi varsa *X* ile *Y* arasında dede-torun ilişkisi vardır. İkinci kurala göreyse *X*'in çocuğu ve *Y*'nin annesi olan bir *Z* kişisi varsa *X* ile *Y* arasında dede-torun ilişkisi vardır.

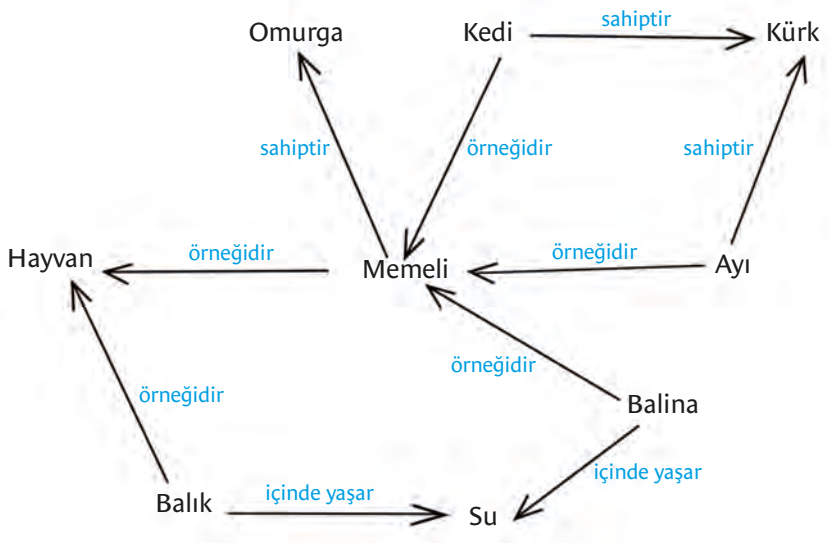
Böyle kodlara sahip bir yapay zekâ uygulamasına Ali'nin dedesinin kim olduğunu öğrenmek için şöyle bir soru ifadesi verilir ve şöyle bir cevap alınır:

? - dede_torun(X, ali).
X = Ahmet

Sembolik yapay zekâ uygulamalarının temel bileşenlerinden biri çıkarım motorudur. Yukarıdaki sonuç, çıkarım motorunun uygulamaya verilmiş gerçekleri ve dede-torun ilişkisi ile ilgili kurallarının ilkini kullanarak çıkarım yapmasıyla ortaya çıkar. Çıkarım motoru, dede-torun ilişkisi ile ilgili ikinci kuralı kullanarak bir cevap üretemez. Çünkü uygulamaya verilen bilgiler yetersiz kalır.

Sembolik yapay zekâ uygulamalarında bilgiyi kodlamak için semantik ağlardan da yararlanılır. Semantik ağlar, yönlü ya da yönsüz grafiklerdir (*bkz.* aşağıdaki grafik). Bu grafiklerin köşelerinde kavramlar yer alır. Grafiklerin kenarları ise kavramlar arasındaki semantik ilişkileri betimler.

Aşağıdaki semantik ağı LISP programlama dilinde kodlamasını istediğimizde ChatGPT yandaki cevabı veriyor. Sağdaki kod dizisinin ilk satırında *semantic-network* isimli bir liste tanımlanıyor. Daha sonra ise listenin elemanları "anahtar-değer" çiftleri olarak sıralanıyor. Bazı değerlerin kendilerinin de listeler olduğunu görüyoruz. Bir yapay zekâ uygulamasından bu listeden balinalar hakkında çıkarım yapması istendiğinde ilk olarak balinalar ile ilgili liste elemanını kullanarak balinaların suda yaşadığını ve memelilerin bir örneği olduğunu bulur. Daha sonra memeliler ile ilgili liste elemanından yaralanarak balinaların bir hayvan olduğu ve bu hayvanların omurgasının olduğu çıkarımını yapar.




```
(defparameter *semantic-
network*
 '(
  (animal
   ((has-subclass . mammal)
    (has-subclass . fish)))

  (mammal
   ((is-a . animal)
    (has . vertebra)
    (has-subclass . cat)
    (has-subclass . bear)
    (has-subclass . whale)))

  (fish
   ((is-a . animal)
    (lives-in . water)))

  (cat
   ((is-a . mammal)
    (has . fur)))

  (bear
   ((is-a . mammal)
    (has . fur)))

  (whale
   ((is-a . mammal)
    (lives-in . water)))

  (vertebra
   ())

  (fur
   ())

  (water
   ())
))
```

Sembolik yapay zekâ uygulamalarının en önemli avantajı insanların okuyup anlayabileceği sembollerle ifade edilmiş, açıkça tanımlanmış kurallara dayalı olmasıdır. Bu durum uygulamanın insan zekâsı gerektiren bir işle ilgili karar verme sürecinin nasıl ilerlediğinin kolayca anlaşılabilmesini sağlar. Ayrıca uygulamadaki hataların düzeltilmesini ve eksikliklerin giderilmesini kolaylaştırır. Örneğin yukarıdaki semantik ağın kodlandığı yapay zekâ uygulaması, balinalar hakkında bilgi vermesi istendiğinde balinaların memeli değil de balık olduğunu söylüyor olsaydı, koddaki hatalı kısmı tespit edip düzeltmek hiç de zor olmazdı. Ya da eğer uygulama papağanlar hakkında bilgi vermesi istendiğinde bir cevap bulamıyorsa yapılması gereken koddaki listeye papağanlar ile ilgili birkaç satır eklemektir.

Sembolik yapay zekâ; kurallara dayalı, yapısal bilginin yer aldığı işlerde ön plana çıkar. İyi tanımlanmış mantık kurallarının ya da süreçlerin yer aldığı işlerde başarılıdır.

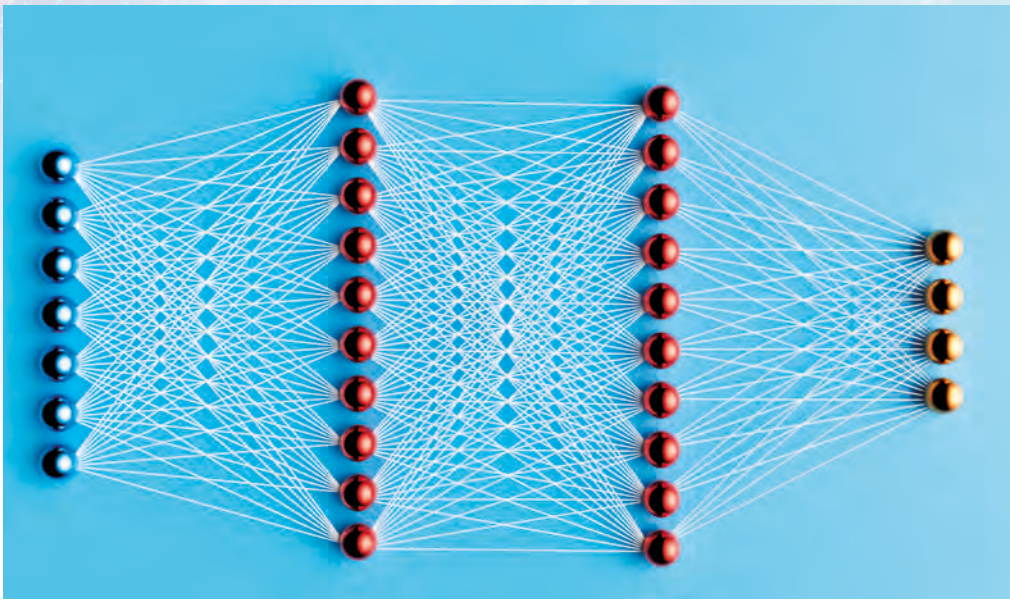
Sembolik yapay zekânın bir zayıf tarafı ölçeklendirmesinin zor olmasıdır. Yapay zekâ uygulamasının bilgi dağarcığı artırılmak istendiğinde kodlanması gereken kuralların sayısı üstel olarak artmaya başlar. Bu durum büyük ölçekli sembolik sistemlerin geliştirilmesini ve idare edilmesini zorlaştırır. Sembolik yapay zekâ görseller, videolar ve yazılı metinler gibi yapısal olmayan veri kaynaklarını ele almakta da pek başarılı değildir. Ayrıca sembolik yapay zekâ uygulamaları insanlar gibi örneklere bakarak ya da inceleyerek yeni bilgi edinmez, bir şeyler öğrenmez. Yeni durumlara uyum sağlamalarının tek yolu bir programcı tarafından yeniden programlanmalarıdır. Sembolik yapay zekâ uygulamalarının bir diğer zayıf tarafı belirsiz, eksik ya da muğlak durumlarla baş edememeleridir. Bu uygulamalar, kodlanmış kesin kuralları kullanarak çıkarım yapabilmek için kesin girdilere ihtiyaç duyar. İnsanlar gibi belirsiz, muğlak durumlar karşısında tavır almaz, bir çıktı üretmezler.

lucadp / iStock

Sembolik yapay zekâ uygulamalarının en başarılı olduğu alanların başında "uzman sistemler" gelir. Sağlık ya da hukuk gibi mesleklerin çalışanlarına yardımcı olmak için tasarlanan bu uygulamalar, kodlarındaki kesin bilgileri ve kuralları kullanarak uzman görüşü bildirir. Örneğin sağlıkla ilgili bir uzman sistem, hastalık belirtilerinden yararlanarak hastalığın ne olduğunun tanısını yapar ve tedavisi için öneriler sunar.

Sembolik yapay zekânın başarılı olduğu bir başka alansa kuralların ve hamlelerin mantığının açıkça ifade edilebildiği satranç gibi strateji oyunlarıdır. Bu konuda akla gelen ilk örneklerden biriyse IBM'in Deep Blue uygulamasıdır.

Satranç kurallarının ve hamlelerinin mantığının sembolik bir biçimde kodlandığı yapay zekâ uygulaması, 1997'de Rus Gary Kasparov'u altı oyunluk bir maçta 3½-2½ yenmiş ve dünya satranç şampiyonu ünvanını elinde bulunduran bir satranç ustasını turnuva koşulları altında mağlup eden ilk yapay zekâ uygulaması olarak tarihe geçmişti.



Thom Leach / SPL

Yapay Sinir Ağları

Yapay zekâ uygulamaları geliştirmek için kullanılan bir diğer programlama stilinde uygulamalar yapay sinir ağlarıyla modellenir. İnsanların sinir sisteminin yapısından esinlenilen bu modellerde yapay nöronlar (sinir hücreleri) olarak adlandırılan birimler bulunur. Biyolojik nöronlar gibi yapay nöronlar da sinapslar aracılığıyla birbiriyle bağlantılıdır. Yapay sinir ağını oluşturan her bir nöron kendisine ulaşan bilgiyi işler ve bağlantılı olduğu diğer nöronlara aktarır.

En basit yapay sinir ağında sadece çeşitli sayıda yapay nörondan oluşan bir girdi katmanı ve yine çeşitli sayıda yapay nörondan oluşan bir çıktı katmanı bulunur. Pratik amaçlar için yararlı olabilecek bir yapay sinir ağı ise genellikle çok katmanlıdır. Derin sinir ağları olarak adlandırılan bu modellerde girdi katmanı ve çıktı katmanı arasında "gizli katmanlar" olarak adlandırılan, çok sayıda yapay nörondan oluşan çeşitli katmanlar bulunur. Yapay sinir ağları ile biyolojik sinir ağları arasındaki benzerlik sadece yapısaldır. Yapay sinir ağları, yapay nöronlar arasında sayısal verilerin aktarılmasıyla çalışır.

Yapay sinir ağlarıyla modellenmiş bir yapay zekâ uygulamasında karşımıza çıkabilecek en basit sinir yapısını ve en basit hesapları ele alalım. Yukarıdaki grafikte betimlenen yapay sinir ağında bir girdi katmanı (mavi) ve bir çıktı katmanının (sarı) yanı sıra iki gizli katman (kırmızı) yer alıyor. Böyle bir sinir ağından "eğitimli olduğu" konularla ilgili bir görevi yerine getirmesi istendiğinde ihtiyacı olan bilgiler "sayılarla kodlandıktan" sonra girdi katmanına verilir. Eğer girdi katmanına sağlanan sayısal verileri, elemanları u_i olan bir u vektörüyle gösterirsek birinci gizli katmandaki yapay nöronların değerlerini içeren bir v vektörü iki aşamada hesaplanır.

İlk olarak $v=w$ x $u+b$ hesabı yapılır. Bu eşitlikteki w, bir matrisi ifade eder. Ağırlık matrisi olarak adlandırılan matrisin elemanları w_{ij} girdi katmanındaki i numaralı yapay nöron ile birinci gizli katmandaki j numaralı yapay nöron arasındaki bağlantıya karşılık gelir, w_{ij} değeri ne kadar büyükse iki nöron arasındaki bağlantı o kadar güçlüdür. Eşitlikteki b vektörü ise hesaba bir miktar "yanlılık" katar, hesaplanan v_j değerlerinin bir miktar kaymasına sebep olur. Birinci aşama tamamlandıktan sonra v vektöründeki nihai değerleri hesaplamak için aktivasyon fonksiyonu olarak adlandırılan bir Φ fonksiyonu v vektörüne uygulanır: $v \rightarrow \Phi(v)$. Yaygın olarak kullanılan çeşitli aktivasyon fonksiyonlarından hangisinin

tercih edileceği, problemin türüne göre değişir ve çoğu zaman hangi fonksiyonun daha iyi sonuç vereceği deneme-yanılma süreçleriyle belirlenir. Aktivasyon fonksiyonunun temel işlevlerinden biri "doğrusallığı bozmaktır". Bu durum yapay zekâ uygulamasının karmaşık problemlerle baş etmesini kolaylaştırır. Birinci gizli katmandaki nöronların değerleri hesaplandıktan sonra sıra ikinci varsa üçüncü, dördüncü ve diğer gizli katmanlardaki ve son olarak çıktı katmanındaki nöronların değerlerinin hesaplanmasına gelir. Bu hesaplar da birinci gizli katmandaki nöronların değerlerinin hesaplanmasına benzer biçimde yapılır. Gizli katmanlardaki nöronlar kendinden önce gelen katmandaki nöronlardan gelen sayısal veriyi işler ve sonuçları bir sonraki katmandaki nöronlara aktarır. Çıktı katmanındaki nöronların sağladığı sayısal değerler, elde edilmek istenen bilgi ile ilgili muhtemel cevapları kodlar.

peterschreiber.media / iStock

Konunun daha iyi anlaşılması için bir örnek ele alalım. Örneğin ikinci el araba fiyatları hakkında tahminler yapan bir yapay zekâ uygulaması olduğunu düşünelim. Uygulamadan bir fiyat tahmini almak için ilk olarak girdi katmanına verilecek bilgilerin sayısal olarak kodlanması gerekir. Fiyatı tahmin edilmek istenen arabanın sadece yaşı ve kilometresi değil, markası, modeli ve fiyatını etkileyebilecek diğer tüm özellikleri de sayısal olarak kodlandıktan sonra derin sinir ağının girdi katmanına verilir. Sinir ağı ağırlık matrislerini, yanlılık vektörlerini ve aktivasyon fonksiyonlarını kullanarak, sayısal verileri bir katmandan diğerine aktararak hesaplar yapar. Böyle bir amaç için tasarlanmış bir sinir ağının çıktı katmanında sadece bir yapay nöron bulunacak ve nöronun sağladığı sayısal veri istediğiniz fiyat tahminini verecektir.

Söz konusu görseller olduğunda da yapay sinir ağları benzer bir biçimde çalışır. Bu sinir ağlarının baş kısmında konvolüsyon katmanları olarak adlandırılan, temel işlevleri görsellerdeki gizli özellikleri tespit etmek olan katmanlar bulunur. Örneğin hayvan resimlerini sınıflandırmak, resimlerin içindeki hayvanların türünü belirlemek için tasarlanmış bir yapay zekâ uygulaması olduğunu düşünelim. Böyle bir uygulamanın sinir ağı girdi olarak uygulamaya verilen resimdeki piksellerin renk kodlarını alır. İlk olarak konvolüsyon katmanları bu sayısal veriyi işleyerek resmin içinde hayvanlara ait özellikler arar. Örneğin resimde kaç tane ayak olduğunu, boynuz ya da kuyruk olup olmadığını belirler. Daha sonra diğer katmanlar hesaplara devam eder. Böyle bir yapay sinir ağının çıktı katmanında her biri farklı türde bir hayvana karşılık gelen çok sayıda yapay nöron bulunur. Hesaplar sonunda bulunan değerler, resimde bir hayvan türünün var olma "olasılığı" hakkında bilgi verir. Örneğin uygulama "Resimde %90 olasılıkla bir kaplan, %7 olasılıkla bir çita, %3 olasılıkla bir aslan vardır." anlamına gelen bir cevap verebilir.

Yapay sinir ağlarının bir görevi başarıyla yerine getirebilmesi sinir ağının yapısına (katmanların türüne, sayısına ve her bir katmandaki nöronların sayısına), hesaplarda kullanılan ağırlık

Olga Tsikarishvili / iStock

matrisi ve yanlılık vektörünün içerdiği sayısal değerlere ve kullanılan aktivasyon fonksiyonlarının türüne bağlıdır. Ancak bu etkenlerin hiçbirisi programcı tarafından önceden kesin olarak belirlenmez. Yapay sinir ağlarına dayalı bir uygulamanın başarılı bir biçimde çalışmaya başlayabilmesi, farklı alternatiflerin hangisinin en iyi sonuç vereceğinin deneme-yanılma ile belirlendiği uzun eğitim süreçleri sonunda mümkün olur.

Yapay sinir ağlarına dayalı uygulamalar her ne kadar kendilerine sunulan örneklerden yeni şeyler öğrenebilseler de insanların öğrenme sürecine kıyasla yapay zekâ uygulamalarının öğrenme süreci hayli verimsizdir. Örneğin az önce bahsettiğimiz hayvan resimlerini sınıflandıran uygulamayı ele alalım. Bir çocuğa bir hayvanın inek mi yoksa koyun mu olduğunu öğretmek için sadece birkaç resim yeterli olsa da bir yapay zekâ uygulamasının bu yeteneği kazanması için gereken örneklerin sayısı binlerle ölçülür. Uygulamanın geliştirilme süreci özetle şu şekilde ilerler: İlk olarak uygulamanın tanıması istenen her bir hayvan türü için binlerce farklı fotoğraf bulunur. Daha sonra bu fotoğrafların bir kısmı (≈%75-80'i) eğitim için geri kalanlarını ise eğitimin başarısını ölçecek testler için ayrılır. Eğitim sırasında uygulamaya hem resimler hem de resimlerdeki hayvanların türünün ne olduğu ile ilgili doğru bilgiler verilir. Uygulama o sırada sahip olduğu ağırlık matrislerini ve vektörleri kullanarak hesaplar yapar. Uygulamanın tahminleri ile doğru sonuçlar arasındaki fark ölçülür. Hata oranını azaltmak için ağırlık matrisleri ve vektörler güncellenir. Tek bir güncelleme hata oranını sıfıra yaklaştırmak için yeterli olmaz. Aynı resimleri kullanarak aynı hesaplar ve güncellemeler binlerce kez tekrar edilir. Ta ki hata oranı arzu edilen derecede düşük bir seviyeye gelene kadar. Daha sonra uygulamanın gerçek dünyada ne kadar iyi sonuç vereceği hakkında fikir edinmek amacıyla testler için ayrılan görsellerle uygulamanın başarısı ölçülür. Eğer uygulama test görselleri için de yüksek oranda doğru sonuçlar verirse eğitimin başarılı olduğuna ve uygulamanın kullanıma hazır olduğuna karar verilir. Eğer test görsellerindeki hata oranı yüksek çıkarsa eğitimin başlangıcında seçilen parametreler değiştirilerek ve hatta belki de farklı bir sinir ağı yapısıyla eğitim sürecine yeniden başlanır. Ta ki kontrolsüz koşullar altında başarılı sonuçlar verecek bir uygulama ortaya çıkana kadar.

Yavaş bir biçimde de olsa eğitilebilmeleri, yapay sinir ağlarına dayalı yapay zekâ programlarının sembolik yapay zekâ programları karşısındaki bir avantajıdır. Ayrıca yapay sinir ağlarının ölçeklendirilmesi daha kolaydır.

Yapay sinir ağları, muğlak durumlarla da baş edebilir, iyi sonuçlar vermek için her zaman kesin bilgiye ihtiyaç duymaz. Örneğin hayvanları sınıflandıran yapay zekâ uygulaması, kendisine verilen görseller bulanık da olsa ya da görsellerdeki hayvanların tüm uzuvları gözükmüyor da olsa yüksek doğrulukla tahminler yapmayı başarabilir.

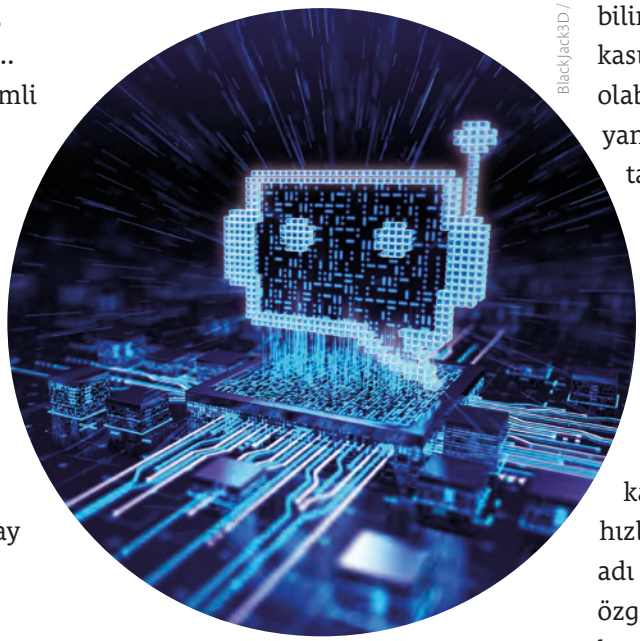
Yapay sinir ağlarının en önemli dezavantajı, uygulamada kodlanmış bilginin insanların okuyup anlayabileceği bir formatta olmamasıdır. Bir yapay sinir ağının hesaplarda kullandığı matrislerin ve vektörlerin içerdiği sayısal

değerlere bakarak uygulamanın bir sonuca varırken hangi adımları takip ettiği ya da nasıl "mantık yürüttüğü" hakkında bir fikir edinmek pek kolay değildir. "Kara kutu sendromu" olarak adlandırılan bu durum yapay sinir ağlarına dayalı uygulamaların hatalardan arındırılmasını zorlaştırır. Eğer uygulama hatalı sonuçlar veriyorsa programcı basitçe sinir ağıyla ilgili sayısal verilere bakıp çeşitli sayıları "düzelterek" hataları gideremez. Yapılabilecek tek şey uygulamanın eğitiminin yetersiz olduğu, daha fazla eğitime ihtiyacı olduğu çıkarımını yapmaktır. Yapay sinir ağlarına dayalı uygulamaların hatalı bilgileri sanki gerçekmiş gibi kullanıcılara sunması yaygındır. Bu durum "yapay halüsinasyon" olarak adlandırılır.

Yapay sinir ağlarının pek çok uygulama alanı vardır: veri işleme, yüz tanıma, görüntü analizi, spam filtreleme, tıbbi teşhis... Yapay sinir ağlarının en önemli uygulama alanlarından biri de üretken yapay zekâdır. Bu uygulamalar eğitimleri sırasında öğrendikleri bilgileri kullanarak yeni metinler, resimler, videolar, müzikler üretir. Büyük dil modelleri (LLM) kapsamına giren ChatGPT, Google Gemini, DeepSeek gibi sohbet robotları da yapay sinir ağlarına dayalı üretken yapay zekâ uygulamalarının örneklerindendir. Bilimsel çalışmalar da yapay sinir ağlarının kullanıldığı alanlardandır. 2024 yılında Nobel Kimya Ödülü'ne konu olan, protein yapılarını tahmin eden AlphaFold programı da yapay sinir ağlarıyla modellenmişti.

Yapay Genel Zekâ

Modern yapay zekâ araştırmalarının başladığı 1950'lerde tüm bilişsel işlerde insanlar kadar hatta belki de daha iyi performans gösterebilecek makineler geliştirmenin, başka bir deyişle "genel yapay zekâ"ya ulaşmanın mümkün olduğu düşünülüyordu. Alanın kurucularından Herbert Simon 1965 yılında "20 sene içinde makinelerin insanların yapabildiği herhangi bir işi yapabileceğini" yazmıştı. Ancak bugün hâlâ yapay genel zekâya ulaşılabilmiş değil. Bu hedefe ulaşabilmenin mümkün olup olmadığı da tartışmalara konu olmaya devam ediyor.



BlackJack3D / iStock

Ne sembolik yapay zekâ uygulamalarının ne de yapay sinir ağlarına dayalı uygulamaların çalışma biçimleri tam olarak insan düşüncesinin ortaya çıkma biçimini taklit etmez: "İkili süreç teorisi", insan düşüncesinin ortaya çıkışında rol alan iki ayrı sistem olduğunu söyler. Bu sistemler sıklıkla "Sistem 1" ve "Sistem 2" olarak adlandırılır. Sistem 1'in bir diğer adı "kapalı sistem"dir. Otomatik bir sürece karşılık gelir. Sürecin otomatik olmasını sağlayan çeşitli bileşenler olabilir. Birincisi, insan meydana gelen zihinsel sürecin farkında olmayabilir. Örneğin süreci tetikleyen uyaran, bilinçaltını etkilemiş olabilir. İkincisi, süreç bilinçli bir şekilde başlatılmamış, kasıtsız bir biçimde gerçekleşiyor olabilir. Üçüncüsü, kontrolsüz yani bir kez başladıktan sonra tamamlanana kadar bilinçli bir biçimde durdurulamayan bir süreç olabilir. Sistem 1 kapsamına giren otomatik süreçlerin en önemli özelliklerinden biri verimliliktir. Otomatik olarak gerçekleşen bir süreç daha az kaynağa ihtiyaç duyar ve daha hızlıdır. Sistem 2'nin bir diğer adı ise "açık sistem"dir. İnsanlara özgü olduğu düşünülen bu sistem, kural tabanlıdır, analitiktir ve

kontrollüdür. Sistem 2, mantık standartlarına göre sıralı bir biçimde mantık yürütür. Sistem 1'e kıyasla çok daha yavaştır.

Sembolik yapay zekâ programlarının çalışması Sistem 2'ye benzer. Kural tabanlıdır, analitiktir, yavaştır. Yapay sinir ağlarının çalışması ise Sistem 1'e benzer. Otomatiktir, hızlıdır fakat iyi tanımlanmış mantık kurallarına dayanmaz.

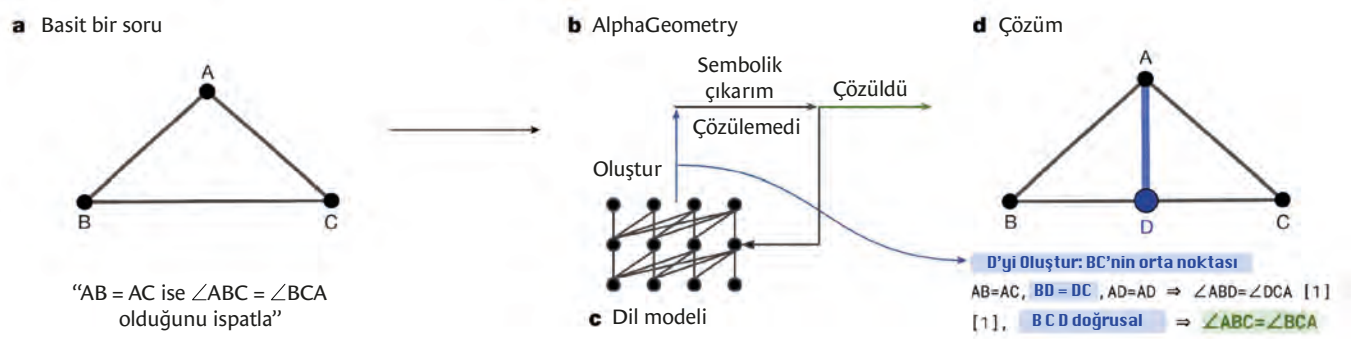
Yapay zekânın gelişmeye başladığı dönemlerde John McCarthy, Herbert Simon ve Allen Newell gibi alanın öncüleri, insan zekâsının sembolik bir biçimde modellenebileceğini düşünmüştü. Ancak sembolik sistemler pek çok konuda yetersiz kaldı. Son zamanlarda ChatGPT gibi yapay sinir ağlarına dayalı uygulamalar önemli başarılar elde etmeye başladı. Fakat bu uygulamalar da tam anlamıyla yapay genel zekâya ulaşmaktan hâlâ çok uzak. Kimilerine göre yapay sinir ağlarının halüsinasyon görmesini engellemenin yolu giderek daha büyük ölçekli (daha çok parametre içeren), daha çok eğitilmiş uygulamalar geliştirmekten geçiyor. Ancak bugün pek çok yapay zekâ araştırmacısı yapay sinir ağlarının da tek başına hiçbir zaman yapay genel zekâya ulaşmak için yeterli olmayacağını düşünüyor. Özellikle son zamanlarda öne çıkmaya başlayan bir yaklaşım ise sembolik programlamayı ve yapay sinir ağlarını bir araya getiren hibrit uygulamalar geliştirmek. Sonuçta insan düşüncesini ortaya çıkaran iki ayrı sistem var, sembolik yapay zekâ bu sistemlerden birine, yapay sinir ağları ise diğerine benziyor. Eğer amaç insanların yapabileceği her işi başaran makineler geliştirmekse insanların sahip olduğu her iki sisteme de sahip makinelerle bu amaca ulaşmaya çalışmak mantıklı görünüyor.

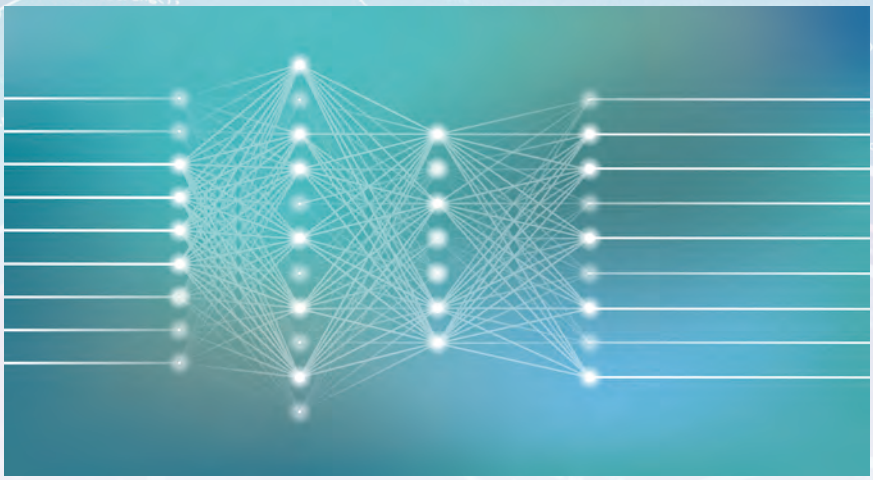
Nörosembolik Yapay Zekâ

Sembolik programlamayı ve yapay sinir ağlarını bir araya getiren yaklaşım "nörosembolik yapay zekâ" olarak adlandırılıyor.

Sembolik algoritmaları ve yapay sinir ağlarını bir araya getirmenin çeşitli yolları var. Bazı nörosembolik uygulamalarda sembolik teknikler, yapay sinir ağlarının performansını artırmak için kullanılıyor. Bu uygulamaların örneklerinden biri AlphaGeometry. Bir büyük dil modeli olan uygulamanın temel işlevi, düzlem geometri ile ilgili teoremler ispatlamak. Uygulama, bir sinir ağının sembolik bir dille üretilmiş sentetik verilerle eğitilmesiyle geliştirilmiş. Dil modeli, zorlu ispat problemlerini çözmesi için bir sembolik çıkarım motorunu yönlendiriyor. Aşağıda AlphaGeometry'nin matematiksel önermeleri nasıl ispatladığının bir örneğini görüyorsunuz.

Programdan $AB=AC$ olmak üzere herhangi bir ABC üçgeninde $\angle ABC=\angle BCA$ olduğunu, başka bir deyişle bir ikiz kenar üçgenin taban açılarının birbirine eşit olduğunu ispatlaması isteniyor. Program ilk olarak sembolik çıkarımla teoremi ispatlamaya çalışıyor ancak başarılı olamıyor. Bu noktada dil modeli devreye





a-image / iStock

girerek *D, BC* doğrusunun orta noktası olacak biçimde bir *AD* doğrusu çizilmesini öneriyor. Böylece program ispatı tamamlamayı başarıyor: İlk olarak *AB=AC,BD=DC,AD=AD* olduğuna göre (benzer üçgenler) *∠ABD=∠DCA* olması gerektiği çıkarımını yapıyor. Daha sonra ise *B,C,D* noktalarının aynı doğru üzerinde yer almasından yola çıkarak *∠ABC= ∠BCA* sonucuna varıyor. AlphaGeometry, orta öğretim öğrencilerinin yarıştığı Uluslararası Matematik Olimpiyatları'nda çıkmış sorularla testten geçirildiğinde altın madalya kazanmış öğrencilere yakın bir performans ortaya koyuyor. Otuz soruluk bir testte 25 soruyu çözmeyi başarıyor.

Nörosembolik yapay zekâ uygulamalarının bir diğeri ise AlphaGo. Go oyunu oynamak için geliştirilmiş bu uygulamada yapay sinir ağları sembolik algoritmaların işini kolaylaştırıyor. Kesin kurallara dayalı bir strateji oyunu olarak sembolik programlaya uygun olsa da Go'daki muhtemel hamlelerin sayısı 10^{170}'in üzerinde. Bir sembolik yapay zekâ programının tüm ihtimalleri tek tek değerlendirip en iyi hamleyi seçmesi uzun zaman alan bir iş. AlphaGo programında olası tüm hamleler tek tek değerlendirilmiyor. Eğitilmiş bir sinir ağı en iyi sonucu verebilecek hamleler hakkında hesaplar yapıyor. Sembolik algoritmalar ise yapay sinir ağının yaptığı az sayıda öneri arasından en iyisini seçiyor. Böylece en iyi hamleye karar vermek için gereken zaman kısalıyor, program çok daha hızlı bir biçimde Go oynayabiliyor.

Bir başka yaklaşım, sembolik algoritmaları büyük dil modellerinin iş akışının ortasına yerleştirmek, mantık yürütme gerektiren kritik aşamalarda sembolik algoritmaların kullanılmasıyla büyük dil modellerin hatalı, istenmeyen sonuçlar üretmesini engellemek. Örneğin PAL (Program-aided Language) modellerinde önce çözülmeye çalışılan problemle ilgili bir Python kodu oluşturmak için büyük dil modelleri kullanılıyor sonra üretilen sembolik kodlarla soru çözülüyor. Son olarak bulunan çözüm büyük dil modeli aracılığıyla yeniden konuşma diline çevriliyor.

Nörosembolik yöntemlerden robot eğitiminde yararlanan araştırmacılar da var. Örneğin J. Mao, J. Tenenbaum ve J. Wu, robotun görüş alanındaki nesneleri tespit etmek için yapay sinir ağlarından, tespit edilen nesneler arasında ilişki kurmak içinse sembolik algoritmalardan yararlanılan bir sistem geliştirdi. İki nesneden hangisinin diğerinin önünde ya da arkasında, sağında ya da solunda olduğunu %99 doğrulukla belirleyecek saf bir yapay sinir ağı modeli eğitmek, yaklaşık 700.000 örnek gerektiriyor. Mao ve arkadaşlarının geliştirdiği nörosembolik sistemi aynı doğruluk oranına ulaştırmak içinse bu sayının sadece %10'u yeterli oluyor. Hatta sadece %1'i bile %92 doğrulukla sonuç veriyor.

Özet

Yapay zekâ araştırmalarının başladığı ilk dönemlerdeki ana yaklaşım sembolik yapay zekâydı. Alanın kurucuları, bilginin ve kuralların sembolik (insanların okuyup anlayabileceği) bir biçimde kodlandığı bu yaklaşımın

yapay genel zekâya ulaşmaya imkân vereceğini düşünüyordu. Sembolik sistemler katı kurallara dayalı, kesin bilgiden yola çıkarak çıkarım yapılan alanlarda başarılı olsa da öğrenmekte ya da belirsiz durumlarla başa çıkmakta zorlanır. Özellikle 2000'lerden sonra ön plana çıkmayan bir başka yaklaşımsa yapay sinir ağları. İnsan beyninin yapısından esinlenerek modellenen bu sistemler, verimsiz bir biçimde de olsa eğitilebiliyor ve hatta belirsiz durumlarla başa çıkmakta da gayet başarılı. Ancak kodlanmış bilginin sembolik ve kesin kurallara dayalı olmaması, yapay sinir ağlarıyla modellenmiş programların kara kutu gibi çalışmasına yol açıyor ve istenmeyen hatalar üretmeleri hâlinde düzeltilmelerini zorlaştırıyor.

Sadece sembolik ve sadece yapay sinir ağlarına dayalı modellerin eksikliklerini gidermek için son yıllarda öne çıkan bir alansa nörosembolik yapay zekâ. Nörosembolik uygulamalar, sembolik algoritmaları ve yapay sinir ağlarını bir araya getiriyor. Sembolik yapay zekânın Sistem 2'ye, yapay sinir ağlarınınsa Sistem 1'e benzemesi, bu hibrit uygulamaların insanlar gibi düşünen yapay zekâ uygulamalarının geliştirilmesi çabasında daha yararlı olacağını düşündürtüyor. AlphaGeometry, AlphaGo gibi yapay zekâ programları, nörosembolik yaklaşımın başarısını matematik ve strateji oyunları gibi sınırlı alanlarda örneklendiriyor. Nörosembolik yaklaşımın bir gün genel yapay zekâya ulaşmasına imkân verip vermeyeceği ya da böyle bir hedefe ulaşmanın mümkün olup olmadığı ise tartışmalara ve araştırmalara konu olmaya devam ediyor.

Bugün var olan nörosembolik uygulamaların çoğunda sembolik algoritmalar ve yapay sinir ağları basitçe bir araya geliyor, belirli görevleri yerine getirmek için "sırayla" çalışıyor. Ancak bu durumun da tam olarak insan düşüncesinin ortaya çıkışını taklit ettiği söylenemez. Bazı araştırmacılara göre yapay genel zekâya ulaşılmasına imkân verecek, sembolik programlama ve yapay sinir ağlarını daha sofistike bir biçimde bir araya getirebilen uygulamalar geliştirebilmek için "yapay zekâ üstbilişi" (yapay zekânın kendi düşünmesini takip ve kontrolü) alanında gelişmelere ihtiyaç var. ■

Just_Super / iStock

Kaynaklar

Jones, N., "The Great AI Mash-up", *Nature*, Cilt 647, s. 842, 2025.
Nawaz, U. ve ark., "A review of neuro-symbolic AI integrating reasoning and learning for advanced cognitive systems", *Intelligent Systems with Applications*, Cilt 26, Makale No: 200541, 2025.
Trinh, T. H., ve ark., "Solving Olympiad geometry without human demonstrations", *Nature*, Cilt 625, s. 476, 2024.
Gao, L., ve ark., "PAL: Program-aided Language Models", *arXiv*, https://arxiv.org/abs/2211.10435, 2022.